



バス構造、データベース 及びインターフェースの設計方法

発明の背景

本発明は、システムLSIなどの半導体集積回路装置中のバスの構造、インターフェースの設計方法及びインターフェースの設計などに用いるためのデータベースに関する。

従来より、半導体回路のCPUとこれによって制御される回路との間を接続するインターフェースと呼ばれる部分は、CPUや各回路間の通信を行なう上で重要な部分である。このインターフェースの中心となる部分は“バス”と呼ばれる信号線であり、データを送る際にはこのバスへのアクセス権をいかに獲得するか、などのデータの入出力を制御するためのシステムが重要である。すなわち、バス構造も含めたインターフェースは最終的なデバイスの性能を大きく左右する要素である。

ここで、図1に示すように、従来のバスの構造としては、ノイマン系プロセッサに用いられるノイマン系バス構造と、ハーバード系プロセッサに用いられるハーバード系バス構造とが知られている。ノイマン系バス構造は、アドレスとデータとのみを区別して、アドレスとデータとを1つの線で表現したり、アドレスとデータとを2つの線で表現する構造のものである。ノイマン系バス構造として、アドレス及びデータを共通のバスを介して流通させるマルチプレクサ型と、アドレスとデータとを別系統のバスつまりアドレスバスとデータバスとを介して個別に流通させるデマルチプレクサ型とが知られている。

また、ハーバード型プロセッサとは、データに対して制御データと実際の転送ファイルのデータとを内容によって分離するための構造である。従来開発されているハーバード系バス構造としては、アドレスバスをさらにI/Oアドレスバスとメモリアドレスバスとに分け、データバスを制御データバスと転送データバスとに分けたもの（以下、これを情報分離型という）が知られている。

マルチプレクサ型バス構造は、アドレス、データの制御と転送とを逐次処理していくために用いられるバス構造であり、これを用いたときの処理能力は比較的

低いがバスの占有面積（バンド幅）は小さい。デマルチプレクサ型バス構造は、アドレスの制御及び転送と、データの制御及び転送とを並列的に処理していくために用いられるバス構造であり、これを用いたときには並列処理が可能なので処理スピードは高くなる。

また、従来のハーバード系バス構造である情報分離型バス構造は、アドレスとデータの双方について、それぞれ制御と転送とを並列的に処理していくために用いられるバス構造であり、これを用いたときの処理能力はさらに高くなる。

ところが、従来のシステムLSIなどの大規模のデバイスを構築する際に、インターフェースの構造をいかに構築するかという点については、未だ適切な手法が確立されていないのが現状である。すなわち、バス構造だけについてみても、各種のバス構造の利点と欠点とがあり、これらを各回路の動作との関連で統一的に評価するための手法は未だ確立されていない。

また、半導体集積回路装置の規模が大きくなり、例えばシステムLSIと呼ばれる複数の半導体回路を組み合わせたデバイスを設計するようになると、上記従来のようなバス構造だけでは、インターフェースを設計する際に、バス構造をフレキシブルに利用できないという不具合もある。

発明の概要

本発明の目的は、システムLSIなどの大規模の半導体デバイスのインターフェースを構築するための設計方法と、その設計に用いるためのバス構造やデータベースを提供することにある。

本発明のバス構造は、半導体集積回路中の制御回路と複数の被制御回路とを接続するためのバス構造であって、上り側バスと下り側バスとに分離されたアドレスバスと、上り側バスと下り側バスとに分離されたデータバスとを備えている。

これにより、ある被制御回路にデータを送りながら他の被制御回路に別のデータを送るような同時命令処理に対する制限が緩和され、バス構造を用いたデバイスのデータ処理能力が向上することになる。

上記バス構造において、上記データバスを、上記被制御回路ごとに分離し、かつ、被制御回路ごとに上り側バスと下り側バスとに分離しておくことにより、さらに同時命令処理に対する制限が緩和される。

本発明のデータベースは、半導体集積回路の設計のために利用されるデータベースであって、制御機能部と複数のアプリケーションとの間を接続するバス構造の種類に関する記述が含まれたテーブルを備えている。

これにより、バス構造によって異なる同時命令処理に対する制限や電力、面積などを総合的に考慮したインターフェースの設計に供しうるデータベースが得られる。

上記データベースにおいて、上記テーブルに、各アプリケーションの動作モデルによって実現される性能を評価するための性能指標を記述した性能テーブルを備えることにより、システム全体の性能の評価を含むインターフェースの構築が可能になる。

上記データベースにおいて、上記性能テーブルは、上記性能指標として、処理能力、バス幅、命令量及びメモリ量のうち少なくともいずれか1つのパラメータを含むことにより、バス構造の種類によって異なるこれらのパラメータを考慮したインターフェースの構築が可能になる。

上記データベースにおいて、上記性能テーブルは、バス構造に関する記述として、上り側バスと下り側バスとに分離されたアドレスバスと、上り側バスと下り側バスとに分離されたデータバスとからなる分離型バス構造に関する記述を含んでいることにより、新たなバス構造を利用したインターフェースの構築が可能になる。

本発明の第1のインターフェースの設計方法は、複数のアプリケーションごとにその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、上記アプリケーションの動作モデルとして上記複数のライブラリを順次用い、上記各ライブラリについて、上記制御機能部により各アプリケーションを無制限に動作させる動作シミュレーションを行なって、バストランザクションの衝突回数を解析する方法である。

この方法により、アプリケーションを動作させたときにバスの混雑度がライブラリの選択によってどう変わるかを考慮して、インターフェースの設計を行なうことが可能になる。

上記第1のインターフェースの設計方法において、各アプリケーション同士の間に上記トランザクションの衝突回数に応じた段数のFIFOを生成し、FIFOを仮挿入した状態で上記解析を行なうことにより、トランザクションの衝突を回避したときの性能を考慮してインターフェースの設計を行なうことが可能になる。

本発明の第2のインターフェースの設計方法は、複数のアプリケーションごとにその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、上記複数のアプリケーションの動作モデルとして上記複数のライブラリを順次用い、上記各ライブラリについて、上記制御機能部により各アプリケーションを無制限に動作させる動作シミュレーションを行なって、同時命令処理の回数を解析する方法である。

この方法によって、アプリケーションを動作させたときのシステムの処理能力がライブラリの選択によってどう変わるかを考慮して、インターフェースの設計を行なうことが可能になる。

上記第2のインターフェースの設計方法において、上記同時命令数に応じて、クロスバスの構造を決定することにより、制御機能部の負荷率の低減や電流値の分散を図ったときの性能を考慮してインターフェースの設計が可能になる。

上記第2のインターフェースの設計方法において、さらに、上記同時命令数が所定値以上であるバスに配置するための転送動作の制御機能部を生成し、転送動作の制御機能部を配置した状態で上記解析を行なうことにより、並列的に転送動作を可能な状態にしたときの性能を考慮したインターフェースの設計が可能になる。

本発明の第3のインターフェースの設計方法は、複数のアプリケーション及び複数のバス構造の各々に対してその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、上記半導体集積回路の価値を最終的に評価するための複数の主パラメータを設定し、上記主パラメータに対して影響を与える複数の副パラメータを設定するステップ(a)と

、上記各主パラメータごとに、各ライブラリの副パラメータによって主パラメータを評価して、主パラメータの目標値を満たすライブラリ群を選び出すステップ（b）と、上記選び出されたライブラリ群ごとに定まる複数の主パラメータを評価して、最適なライブラリ群を選ぶことにより、インターフェースを決定するステップ（c）とを含んでいる。

この方法により、一度にすべてのパラメータから性能を評価するのに比べ、より統一的な手法で最適なライブラリ群を選び出し、最終的に最適なインターフェースを決定することができる。

上記第3のインターフェースの設計方法において、上記ステップ（a）の前に、上記複数のアプリケーションの動作モデルとして上記複数のライブラリを順次用いて、上記各ライブラリについての動作シミュレーションを行なうことにより、より正確に最適なインターフェースを決定することができる。この方法の具体的な手法としては、以下ののような手法がある。

例えば、上記ステップ（a）では、3つの主パラメータを設定し、各主パラメータごとに3つの副パラメータを設定して、上記ステップ（b）では、3つの副パラメータを座標軸とする3次元座標系を構成して、各副パラメータの値によって定まる3角形の面積が目標値以下のライブラリ群を選び出し、上記ステップ（c）では、3つの主パラメータを座標軸とする3次元座標系を構成して、選び出されたライブラリ群から求まる主パラメータの値によって定まる3角形の面積が最小のライブラリ群からインターフェースを決定することができる。

また、上記ステップ（a）の後ステップ（b）の前に、上記複数の副パラメータのうち特定の副パラメータに着目して、特定の副パラメータが目標値を満たしているライブラリ群を選び出し、上記ステップ（b）では、上記複数の主パラメータのうち特定の主パラメータを除く主パラメータが目標値を満たしているライブラリ群を選び出し、上記ステップ（c）では、上記特定の主パラメータが最小となるライブラリ群を最適なライブラリ群として選び出することもできる。

さらに、上記ステップ（a）では、主パラメータに対する複数の副パラメータの影響係数を設定し、上記ステップ（b）では、影響係数と副パラメータの値に基づいて各主パラメータの目標値を満たすライブラリ群を選び出し、上記ステ

ップ (b) では、選び出されたライブラリ群から求まる複数の主パラメータに重み付けを行なってから、ライブラリ群を決定するための演算を行なうこともできる。

本発明の第4のインターフェースの設計方法は、複数のアプリケーション及び複数のバス構造の各々に対してその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、上記複数のアプリケーションの動作モデルとして上記複数のライブラリを順次選択するステップ (a) と、上記制御機能部により上記各アプリケーションを動作させて、各ライブラリを用いたときの制御機能部、インターフェース及びアプリケーションの性能を解析するステップ (b) と、上記ステップ (a) とステップ (b) とを繰り返し行なって、上記解析結果に基づいて最適なライブラリ群を選ぶことにより、インターフェースを決定するステップ (c) と、上記決定されたパラメータに基づいて最適なインターフェースを合成するステップ (d) とを含んでいる。

この方法により、各アプリケーションを動作させたときのシステム全体の性能を評価した上で、最適なインターフェースを合成することが可能になり、インターフェース設計の基本的な手法を確立することができる。

上記第4のインターフェースの設計方法において、上記ステップ (b) では、上記各ライブラリについて、上記制御機能部により各アプリケーションを無制限に動作させたときのバストランザクションの衝突回数を解析し、上記ステップ (d) では、各アプリケーション同士の間に上記トランザクションの衝突回数に応じた段数の FIFO を挿入することが好ましい。

上記第4のインターフェースの設計方法において、上記ステップ (b) では、上記各ライブラリについて、上記制御機能部により各アプリケーションを無制限に動作させたときの同時命令処理の回数を解析し、上記ステップ (d) では、上記同時命令数が所定値以上であるバスにはクロスバスを配置することをが好ましい。

図面の簡単な説明

図1は、従来のノイマン系バス構造、従来のハーバード系バス構造及び本発明

の第1の実施形態のハーバード系バス構造の構造上の相違を概略的に示すブロック図である。

図2(a)～(c)は、従来のバス構造、第1の本実施形態の方向分離型バス構造(3次局がある場合及びない場合)の例を順に示す図である。

図3(a)～(d)は、順に、ノイマン系のマルチプレクサ型、デマルチプレクサ型、ハーバード系の情報分離型及び第1の実施形態の方向分離型バス構造におけるアドレス及びデータの処理を時間軸で示す図である。

図4は、第1の実施形態における対象リソースAがメモリで、対象リソースB、CがI/Oである場合のリソース分離型バス構造を示すブロック図である。

図5(a)、(b)は、第2の実施形態における設計用のデータベースに記憶されている単体アプリケーションA、Bでのライブラリ及び性能テーブルの例を示す図である。

図6は、第2の実施形態における複数のアプリケーションを用いて動作シミュレーションを行なう方法の例を示すブロック図である。

図7は、第2の実施形態における性能解析のうちのトランザクション解析の表示方法を示す図である。

図8は、第2の実施形態における性能解析のうちの命令処理解析の表示方法を示す図である。

図9(a)、(b)は、第2の実施形態において同時命令の多い箇所にクロスバスを設けた場合のバス構造の一部を示す図である。

図10(a)～(d)は、各パラメータが目標値を満たしているライブラリの中で交差面積が最小のものを選択する方法の手順を示す図である。

図11(a)～(d)は、性能指数、平均電力指数(又は最大電力指数)及び面積指数についてそれぞれ重み付け行なってから加算し、その加算値である最適指数が最小のものを選択する方法である。

図12は、第2の実施形態のインターフェースの設計方法を用いて合成される最適IFの構造を示すブロック図である。

図13は、第2の実施形態における最適IFの合成を含むシステム設計の手順を示すフローチャート図である。

好適実施形態の説明

(第1の実施形態)

図1は、従来のノイマン系バス構造と、従来のハーバード系バス構造と、本実施形態のハーバード系バス構造との構造上の相違を概略的に示すブロック図である。

本実施形態におけるハーバード系バス構造は、“方向分離型バス構造”というべきものであって、情報分離型バス構造における転送データをさらに分離して処理するものといえる。この方向分離型バス構造は、プロセッサ（制御回路）を中心として対象リソース（被制御回路）を配置したときの転送データが送られる方向を、“上り”と“下り”とに分けるものであるが、さらに、転送データの被制御回路をメモリ用データとI/O用データとに分ける場合もある。ここで、制御データとは、例えば認識、応答などのコントロールの情報を持ったものである。転送データとは、例えば画像ファイルのデータなどのまとまったデータをいう。

図2(a)は、従来のバス構造を示す図である。同図に示すように、従来のバス構造では、CPUと対象リソース（ここでは1次局I/O）との間には1つのバスしか設けられていなかった。

それに対し、本実施形態の方向分離型バス構造においては、図2(b), (c)に示すように、プロセッサ(CPU)と一次局との間の通信をも、上り側バスと下り側バスとに分けて行なうように構成されている。ここで、図2(b)は、各々1次局を挟んでCPUと通信を行なう2次局と3次局とが存在している場合の方向分離型バス構造を示す図であり、図2(c)は、図2(b)における3次局が存在しない場合の方向分離型バス構造を示す図である。

図3(a)～(d)は、順に、ノイマン系のマルチプレクサ型バス構造、ノイマン系のデマルチプレクサ型バス構造、ハーバード系の情報分離型バス構造及び方向分離型バス構造（方向型）におけるアドレス及びデータの処理を時間軸で示す図である。

図3(a)に示すように、マルチプレクサ型バス構造は、アドレスとデータとをいわば1つの線上でシリアルに処理していくためのバス構造である。例えば対象リソースAに対して何らかのコマンドを生成する場合に、対象リソースAの制

御アドレスを指定し、対象リソース A の制御データを送り、対象リソース A に転送するためのアドレスを送り、次に A への転送データを送る。“B”に対するコマンドを生成する際にも、同様の手順による処理を逐次（シリアルに）行なうのである。

図 3 (b) に示すように、デマルチプレクサ型バス構造は、アドレスとデータとを並列的に処理していくためのバス構造である。例えば対象リソース A に対して何らかのコマンドを生成する場合に、対象リソース A の制御アドレスを指定しながら A の制御データを送り、A に転送するためのアドレスを送っている間に対象リソース A への転送データを送る。対象リソース B に対するコマンドを生成する際にも、同様の手順による処理を並列的に行なうのである。

図 3 (c) に示すように、情報分離型バス構造は、制御アドレス、転送アドレス、制御データ及び転送データをそれぞれ並列的に処理していくためのバス構造である。例えば対象リソース A に対して何らかのコマンドを生成する場合に、対象リソース A の制御アドレスの指定と、対象リソース A への制御データの送信と、対象リソース A に転送するためのアドレスの送信と、対象リソース A への転送データの送信とを並列的に処理する。対象リソース B に対するコマンドを生成する際にも、同様の手順による処理を並列的に行なう。

図 3 (d) に示すように、情報分離型バス構造は、制御アドレス、転送アドレス、制御データ及び転送データをそれぞれ並列的に処理していくに加えて、アドレス及びデータの転送を、上りと下りとに分けて並列的に行なうためのバス構造である。例えば対象リソース A, B, C に対して何らかのコマンドを生成する場合に、対象リソース A, B, C の制御アドレスを指定するとともに上りアドレスと下りアドレスとも指定しておく。そして、上りアドレスを指定しているときには上り側バスを介して上り側の転送アドレス及び転送データ（例えば対象リソース A, C についてのもの）の送信を、下りアドレスを指定しているときには下り側バスを利用して下り側の転送アドレス及び転送データ（例えば対象リソース B, A についてのもの）を送信することができる。言い換えると、制御アドレスの指定や制御データの送信とは無関係に、転送アドレスの送信と転送データの送信とを行なうことができる。

例えば、図2（b）に示すバス構造を有する場合、1次局IO1が対象リソースAで、1次局IO2が対象リソースBで、2次局が対象リソースCであるとする。このとき、対象リソースA（1次局IO1）のアドレスを指定している際にも、対象リソースB（1次局IO2）には対象リソースA（1次局IO1）のアドレスの指定が入力されないので、転送アドレスAを指定して転送データを対象リソースAに転送している間に、転送アドレスBを指定して転送データを対象リソースBに送ることが可能になる。そして、データをインプット（上り）すると、すぐにデータをアウトプット（下り）することが可能になる。

図4は、対象リソースAがメモリで、対象リソースB、CがIOである場合のリソース分離型バス構造を示すブロック図である。対象リソースC（IO2）からのデータの入力を受け、対象リソースA（メモリ）にこれを記憶させながら、対象リソースB（IO1）にデータを転送するという動作が同時に見えることになる。

その結果、従来の情報分離型バス構造を用いた場合には、データ転送に要する時間は、図3（d）に示すように、データ転送A、データ転送B、データ転送C及びデータ転送Aに要する時間をシリアルに加算したものになる（転送A、B、C、Aアドレスの時間も同じ）が、本実施形態の方向分離型バス構造を用いた場合には、図3（d）に示すように短縮されることになる。つまり、上り側バスと下り側バスとに分けることによって、対象リソースA、Cに対する上り側の転送アドレス及び転送データの送信と、対象リソースB、Aに対する下り側の転送アドレス及び転送データの送信とを同時に行なうことが可能になるからである。

（第2の実施形態）

次に、バス構造をも含めたIFの設計方法に関する第2の実施形態について説明する。ここでは、最適なIFを設計する手順について説明する。

－準備するデータ－

ここでは、まず性能解析を行なうために必要なデータ（ライブラリモデル）について説明する。単体アプリケーションごとに、動作モデルを作成する必要があるが、その動作モデルの1つの作成方法として、基本的にソフトウェアで作成し、あるものはここまでをハードウェアで作成し、他のものは別の部分までをハー

ドウェアで行なうように決定する。なお、単体アプリケーションとは、1つのアプリケーションの入出力関係を定義したもので、範囲の大きさは定義していない。例えば、単体アプリケーションの例として、IrDAを用いたプリントデータ転送の動作や、USBを用いたマウスの位置データの転送や、JPEGを用いた静止画像データの圧縮、解凍動作などがある。ここで、“IrDA”と呼ばれる赤外線通信を用いてデータの転送についてのアプリケーションを使用するときには、コンピュータ側で例えばプリントするデータを事前に加工しなければならない。例えば、静止画を圧縮するためにJPEGで圧縮する加工を施して、そのデータを赤外線として表現するためのフォーマット変換を行なってから、データを送るという赤外線通信を行なうことになる。その場合に、ここでいう単体アプリケーションとは、赤外線通信（IrDA）を行なう前の処理は除いて、IrDAと呼ばれているアプリケーションの入力と出力を規定している部分だけを意味している。

ここで、動作モデルをハードウェアで作成するか、ソフトウェアで作成するかによって処理時間が異なってくる。本実施形態では、1つのプロトコルの表現において各層がある程度明確に分かれている部分で、動作モデルをハードウェアで作成するかソフトウェアで作成するかを分ける。

図5(a), (b)は、設計用のデータベースに記憶されている単体アプリケーションA, Bでのライブラリ及び性能テーブルの例を示す図である。例えば、アプリケーションA, Bに対して、仕様レベルのライブラリA, Bがあるとすると、動作レベルにおいては、同図に示すように、アプリケーションA, Bごとに、CPU(バス構造)として、ノイマン型CPU(バス構造), 従来のハーバード型CPU(バス構造), 方向分離型CPU(バス構造)を採用したときの動作を記述した動作モデルとなるライブラリが登録されている。そして、各ライブラリの性能テーブルにおいては、各ライブラリの性能指標が、処理能力(P), バス幅(B), 命令量(M), メモリ量(E)という変数(パラメータ)の関数として表されている。

図5(a), (b)において、ライブラリの動作を記述した部分のうちハッキング部分がソフトウェアによって実現する部分を、白地の部分がハードウェアに

よって実現する部分をそれぞれ示している。例えばアプリケーションAにおいて、各機能のすべてをソフトウェアで実現すると性能指数が100になるとすると、ハードウェアで40%をソフトウェアで残りの60%をそれぞれ実現させる場合には、例えば性能指数が50になり、ハードウェアで60%を実現した場合には、性能指数が10になる。このように、ソフトウェアのうちの何%をハードウェアで置き換えると性能指数がいくらで済むかについての性能テーブルが準備されている。

ここで、動作モデルの機能のうち、“FLOW”とは、例えばa, bという処理がある場合に、“処理aを行なってから処理bを行なう”というような処理のフローを記述した層をいう。“MANG”とは、例えば、別々のファイル転送アプリケーション同士が接続されているときに、その各々に必要なデータ交換のマルチプレクサなどのアプリケーション間の通信をマネジメントする方法を記述した層をいう。“LINK”とは、例えば1つの情報データが、同期ビット、コントロールデータ、MACデータ、終了などの一連のデータによって囲まれた範囲をデータとして認識させるというような、接続の手順を定めている層をいう。“PHY”とは、実際のコーディングの仕方、例えば“1”を表現するときに、あるパルス幅においてすべて“1”になっているか、あるパルス幅の中央部で“1”になっているか、などを区別している層をいう。また、“CALL”とは、例えば演算処理の場合には、演算処理（かけ算など）をハードウェアで行なうのか、ソフトウェアで行なうのかを示す層をいう。

また、ソフトウェアで行なう場合には高い性能が必要となるが、ハードウェアがある程度あることによって性能が低くても済むことが多いことに留意しておく必要がある。そして、図5(a), (b)に示すような性能テーブルが、多数のアプリケーションについて準備されているが、図5(a), (b)にはアプリケーションA, Bのみが例示されている。

－動作シミュレーション－

図6は、複数のアプリケーションを用いて動作シミュレーションを行なう方法の例を示すブロック図である。ここでは、アプリケーションAには100%ソフトウェア化したライブラリを、アプリケーションBは20%だけハード化した

ライブラリ、アプリケーションCには40%だけハード化したライブラリを用いると仮定して、この3つのアプリケーションの転送処理の性能解析を行なうとする。

ここで、図6に示す接続関係において、インプット、アウトプットをカウントする部分（出入力カウント部）と、命令処理をカウントする部分（命令処理カウント部）とを、動作解析シミュレーション中に記述しておく。出入力カウント部とは、例えば、転送処理айл（システム動作制御）に対してアプリケーションA, B, Cが接続されている場合に、何も処置を施さずに転送処理を行なったとすると、プロセッサと各機器との間における入出力が互いに衝突（バストランザクションの衝突）する回数をカウントする。つまり、バスの混雑度を測定するのである。また、命令処理がどのくらいの数だけ発生しているかを、その衝突も含めてカウントしておく。

図7は、この性能解析のうちのトランザクション解析の表示方法を示す図である。同図に示すように、各アプリケーションA, B, Cについてノイマン型バス構造（デマルチプレクサ型）のライブラリを採用した場合と、従来のハーバード型バス構造（情報分離型）のライブラリを採用した場合と、方向分離型バス構造のライブラリ採用した場合について、出入力カウント部で測定されたアプリケーションA-B間、アプリケーションB-C間、アプリケーションA-C間のトランザクション密度を処理時間軸の方向にまとめる。図7において、ハッチングの密度が濃いほど衝突回数が多くなるものとする。ノイマン型バス構造を採用した場合は、並列処理を行なうとどうしても衝突が多くなることがわかる。それに対し、方向分離型バス構造を採用した場合は、並列処理が容易な構造であるので、衝突回数は比較的少なくなる。そして、この衝突回数の多い箇所にFIFOを挿入する。例えば、ノイマン型バス構造を採用する場合には、A-B間にFIFOをk段（例えば10段）挿入し、B-C間にFIFOを1段（例えば8段）挿入し、A-C間にFIFOを1段挿入する。また、従来のハーバード型バス構造（情報分離型）を採用した場合には、A-B間にFIFOを1段挿入し、B-C間にFIFOをm段（例えば6段）挿入し、A-C間にFIFOをm段挿入する。また、方向分離型バス構造を採用した場合には、A-B間にFIFOをn段（

例えば4段)挿入し、B-C間にFIFOをn段挿入し、A-C間にもFIFOをn段挿入する。CPUが処理している間に、FIFOで情報をためておく処理が必要となるが、バスの混雑度に応じてFIFOの段数を解析しておくのである。もちろん、バストランザクションの衝突回数が少ないほど、挿入すべきFIFOの段数は少なくなる。このようにして、それぞれのバス構造の構造つまりCPUにつながるバスの構造に応じて、FIFOを何段挿入する必要があるかが解析される。

次に、図8は、この性能解析のうちの命令処理解析の表示方法を示す図である。同図に示すように、アプリケーションA, B, Cについて、ノイマン型バス構造(デマルチプレクサ型)のライブラリを採用した場合と、従来のハーバード型バス構造(情報分離型)のライブラリを採用した場合と、方向分離型バス構造のライブラリを採用した場合について、命令処理カウント部で測定されたアプリケーションA-B間、アプリケーションB-C間、アプリケーションA-C間の同時命令密度を処理時間軸の方向にまとめる。図8において、ハッチングの密度が濃いほど同時命令回数が多くなり、システム全体の処理能力が高いことを示している。ノイマン型バス構造を採用した場合は、同時命令回数が少ないとわかる。それに対し、方向分離型バス構造を採用した場合は、同時命令回数が比較的多くなる。同時命令回数が多い場合には、処理時間が短くなり応答スピードが速くなるが、反面、CPUの負荷率が増大し、瞬間的な電流値も増大するという不具合もある。

図9(a), (b)は、この同時命令の多い箇所にクロスバスを設けた場合のバス構造の一部を示す図である。図9(a)に示すように、CPUの機能に余力がある場合には、通常使用していない部分にアプリケーションBの転送機能を持たせるようにしておいて、CPUによりクロスバスの切り替えを制御する。あるいは、図9(b)に示すように、CPUに余力がないなどの場合には、DMAを配置する。このDMA(ダイレクト・メモリ・アクセス)とは、入出力コントローラと主記憶との間でCPUを介さず直接データをやりとりさせるという転送機能を持ったものである。すなわち、CPUですべてを行なうのではなく、DMAという機能を利用し、スイッチング機能を有するクロスバスを設けることで、例

えばアプリケーションAはCPUで、アプリケーションBはDMAで処理するような切り替えを行なうことにより、CPUの負荷率の低減と電流値の分散とを図るのである。この新たに配置するものは転送機能さえ有していればよいのであるが、DMAの代わりにCPUを配置してもよいことはいうまでもない。そして、同時命令回数がもっとも多い部分に着目し、その型のCPUを採用したときにおける同時命令回数の最高値に応じて、クロスバスを設けないか、一重クロスバスを設けるか、二重クロスバスを設けるかを選択する。

－ 解析指標－

次に、以上の解析結果で得られる性能のうち最適なものを決定する方法について説明する。本実施形態の解析方法においては、システムの価値を決定する最も重要なパラメータを主パラメータとする。ここでは、性能、電力、面積を主パラメータとする。そして、これらの主パラメータに影響を与える3つの副パラメータを総合し、各主パラメータが目標とする範囲に収まるライブラリ群をまず選び出し、選び出された各ライブラリの主パラメータの値に基づいて、総合的に最適なライブラリ群を選択する方法である。

以下、具体的な方法について順に説明する。

1. 交差面積の最小のものを選択する方法

図10(a)～(d)は、各パラメータを満たしているものの中で交差面積が最小のものを選択する方法の手順を示す図である。

まず、図10(a)に示すように、“性能”という共通の主パラメータに影響を与える副パラメータとして、バストランザクションの衝突回数、処理量及び応答時間という3つの副パラメータを設定し、この副パラメータを座標軸とする3次元の座標系を作成する。この3つの副パラメータの値は、各アプリケーションの動作モデルである各ライブラリを用いて動作シミュレーションを行なった解析結果から求められる。例えばCPUに接続される単体アプリケーションA, B, Cの構成を、ノイマン型バス構造を採用するか、従来のハーバード系バス構造を採用するか、方向分離型バス構造を採用するかということと、さらに、各アプリケーションの動作におけるハードウェアの割合をどのように設定するかによって、一意的に定まる。例えば、図7, 図8に示す解析結果から、バストランザクシ

ヨンの衝突回数や処理時間が求まる。このとき、各 I/O 間のトランザクションの衝突回数の最悪値もしくは平均値、あるいは全体の平均値、ある区間の平均値などをトランザクション T の点とすることができます。応答時間 R は、図 7、図 8 に示すデータ解析を行なったときに実行（シミュレーション）時間である。処理量 E は、図 7、図 8 に示す解析を行なったときの実行処理量（合計値）である。

そして、バストランザクション T が少ないほど、処理量 E が少ないほど、応答時間 R が短いほどシステム LSI の性能が高いことから、3 次元座標系における交差面積が小さいほど性能という主パラメータの 1 つがベターになる。そこで、この座標系における各座標軸上の各点（副パラメータの値）を結んで形成される平面と座標軸との交差面積がある目標値以下となるライブラリ群を選別しておく。そして、この交差面積を相対的には“性能”の値としても差し支えない。

また、図 10 (b) に示すように、“電力”という共通の主パラメータに影響を与える 3 つの副パラメータとして、処理量 E と、ハードウェア化率 H と、同時アクティブ密度 A とを設定し、この 3 つの副パラメータを座標軸とする 3 次元の座標系を作成する。ハード化率 H は、図 6 に示すライブラリモデルにおけるハードウェアの割合である。同時アクティブ密度 A は、図 7、図 8 に示す解析における同時に活性になっているバスの割合である。このときには、処理量 E が少ないほど、ハードウェア化率 H が小さいほど、同時アクティブ密度 A が小さいほど電力は少なくて済むので、3 次元座標系における交差面積が小さいほど電力という主パラメータがベターになる。そこで、この座標系における各座標軸上の各点（副パラメータの値）を結んで形成される平面と座標軸との交差面積がある目標値以下となるライブラリ群を選別しておく。そして、この交差面積を相対的には“電力”の値としても差し支えない。ただし、この方法には、電力の最大値（ピーク値）で評価するか、電力の平均値で評価するかの 2 通りの方法があり、いずれでもよいものとする。

さらに、図 10 (c) に示すように、“面積（コスト）”という共通の主パラメータに影響を与える 3 つの副パラメータとして、必要なバス幅 B と、必要なメモリ量 M と、挿入しなければならない FIFO 量 F とを設定し、この副パラメータを座標軸とする 3 次元の座標系を作成する。バス幅 B は、図 6 に示すモデルに

おけるバス構造におけるバスの合計幅である。メモリ量Mは、図6に示すライブラリモデルで使用するメモリ量である。FIFO量Fは、図7、図8に示す解析の結果必要とされるFIFO段数の合計である。このとき、バス幅Bが小さいほど、メモリ量Mが少ないほど、FIFO量Fが少ないほど面積という主パラメータがベターになる。そこで、この座標系における各座標軸上の各点（副パラメータの値）を結んで形成される平面と座標軸との交差面積がある目標値以下となるライブラリ群を選別しておく。そして、この交差面積を相対的には“面積”の値としても差し支えない。

そして、図10(d)に示すように、主パラメータ、つまり、性能、電力、面積を座標軸とする3次元座標系を作成する。そして、図3(a)～(c)の処理によって選び出された各ライブラリ群によってそれぞれ定まる主パラメータの値、つまり、性能、電力、面積の値（図10(a)～(c)の交差面積）を示す座標軸上の各点を結んで形成される平面と座標軸との交差面積（3角形の面積）が最小となるライブラリ群を選び出し、このライブラリ群によって決定されるインターフェースを最適なインターフェースとして決定する。

2. 特定のパラメータを重視して選択する方法

例えば、図7、図8に示す解析結果から、副パラメータである目標応答時間満たしているライブラリ群の中で、副パラメータであるバストランザクションが所定値以下のもの、副パラメータである目標応答時間を満たしているライブラリ群の中で副パラメータである処理量が所定値以下のもの、副パラメータである目標メモリ量を満たしているライブラリ群の中で副パラメータであるバス幅が所定値以下のもの、副パラメータである目標FIFO量を満たしているライブラリ群の中で副パラメータであるバス幅が所定値以下のものを選択する。

次に、上記の処理によって選び出されたものの中から、さらに以下の基準によって総合的に最適と思われるものを決定する。この処理には、設計対象の種類等に応じて以下のような方法がある。

第1は、選び出されたライブラリ群の中から、性能、最大電力という主パラメータが目標最低性能、目標最大電力を満たしているものを選別する。そして、その中で主パラメータである面積が最小のライブラリ群を選択する方法である。

第2は、選び出されたライブラリ群の中から、性能、電力という主パラメータが目標最低性能、目標最大電力を満たしているものを選別する。そして、その中で主パラメータである平均電力が最小のライブラリ群を選択する方法である。

第3は、選び出されたライブラリ群の中から、面積、最大電力という主パラメータが目標最大面積、目標最大電力を満たしているものを選別する。そして、その中で主パラメータである性能が最小のライブラリ群を選択する方法である。

第4は、選び出されたライブラリ群の中から、面積、最大電力という主パラメータが目標最大面積、目標最大電力を満たしているものを選別する。そして、その中で主パラメータである平均電力が最小のライブラリを選択する方法である。

3. 重み付けを行なって指数化する方法

この方法は、図11(a)～(d)に示すように、主パラメータである性能に対する性能指数 x 、主パラメータである電力に対する平均電力指数 y_{av} （又は最大電力指数 y_{mx} ）、主パラメータである面積に対する面積指数 z についてそれぞれ重み付け a 、 b 、 c を行なってから加算し、その加算値である最適指数が最小のものを選択する方法である。

ここで、性能指数 x は、以下の手順により算出する。つまり、図11(a)に示すように、応答時間 R 及びその性能影響係数 1_x と、バストランザクション T 及びその性能影響係数 m_x と、処理量 E 及びその性能影響係数 n_x とを算出し、下記式

$$\text{性能指数 } x = R 1_x \times T m_x \times E n_x$$

により、性能指数 x を算出する。ただし、応答時間の性能影響係数 1_x は、例えば1 secのときを“1”とする。つまり、応答時間が3 secのときには応答時間の性能影響係数 1_x は“3”となる。バストランザクションの性能影響係数 m_x は、10回の衝突があるときを“1”とする。つまり、20回の衝突があるときには、バストランザクションの性能影響係数は“2”となる。処理量の性能影響係数 n_x は、10 MIPSのとき（1 MIPSは100万回の命令数を意味する）を“1”とする。つまり、50 MIPSのときには、処理量の性能影響係数 n_x は“5”となる。

電力指数 y は、以下の手順により算出する。つまり、図11(b)に示すよう

に、平均処理量 E_{av} (又は最大処理量 E_{mx}) 及びその電力影響係数 l_y 、と、ハード化率 H 及びその電力影響係数 m_y 、と、平均同時アクティブ率 A_{av} (又は最大同時アクティブ率 A_{mx}) 及びその電力影響係数 n_y 、とを算出し、下記式

$$\text{電力指数 } y = E_{av} l_y \times H m_y \times A_{av} n_y$$

$$O_r = E_{mx} l_y \times H m_y \times A_{mx} n_y$$

により、電力指数 y を算出する。ただし、平均処理量 (又は最大処理量) の電力影響係数 l_y は、例えば 10 MIPS のときを “1” とする。ハード化率の電力影響係数 m_y は、20% のときを “1” とする。つまり、40% のハード化率のときには、ハード化率の電力影響係数は “2” となる。平均同時アクティブ率 (又は最大同時アクティブ率) の電力影響係数 n_y は、25% のときを “1” とする。つまり、10% のときには、平均同時アクティブ率の電力影響係数 n_y は “0.5” となる。

面積指数 z は、以下の手順により算出する。つまり、図 11 (b) に示すように、メモリ量 M 及びその面積影響係数 l_z 、と、FIFO 量 F 及びその面積影響係数 m_z 、と、バス幅 B 及びその面積影響係数 n_z 、とを算出し、下記式

$$\text{面積指数 } z = M l_z \times F m_z \times B n_z$$

により、面積指数 z を算出する。ただし、メモリ量の面積影響係数 l_z は、例えば 1 kByte のときを “1” とする。つまり、10 kByte のときにはメモリ量の面積影響係数 l_z は 10 である。FIFO 量の面積影響係数 m_z は、128 Byte のときを “1” とする。つまり、256 Byte のメモリ量のときには、メモリ量の面積影響係数は “2” となる。バス幅の面積影響係数 n_z は、16 bit のときを “1” とする。つまり、8 bit のときには、バス幅の面積影響係数 n_z は “0.5” となる。

そして、図 11 (d) に示すように、解析指標の判断は、上述の処理によって算出された性能指數 x 、電力指數 y 、面積指數 z について、それぞれの影響係数 (重み付け係数) a 、 b 、 c を決定し、それを加算した値を最適指數とする。つまり、下記式

$$\text{最適指數 } O_p = a x + b y + c z$$

に基づき、最適指數 O_p を算出する。そして、最適指數 O_p が最小のものを最終

的に選択するのである。

－最適 I F 合成－

図 12 は、上述の処理を用いて合成される最適 I F の構造を示すブロック図である。上述の解析指標に基づいて選択されたバス構造において、FIFO を挿入する必要のある箇所には FIFO を挿入し、同時処理可能なバス構造を生成するのである。なお、同図において、DMA を除いて一点鎖線で囲まれた部分が I F (インターフェース) の部分である。すなわち、データバス、制御バス、クロスバス等を含むバス構造と挿入された FIFO とにより、ライブラリ A, B, C のハードウェアと、CPU (動作モデル) と、記憶装置 (RAM, ROMなど) とを接続する I F が構成されている。

図 13 は、本実施形態において説明した最適 I F の合成を含むシステム設計の手順を示すフローチャート図である。

まず、データベースとして、図 5 に示すライブラリ A, B, C や、第 1 の実施形態で説明したハーバード型バス構造、ノイマン型バス構造 (デマルチプレクサ型)、方向分離型バス構造などを記憶しておく。

そして、ステップ ST 1 で、選択されたライブラリに対してバス構造を選択し、ステップ ST 2 で、システム接続図、転送処理ファイル、新規ライブラリ等を入力する。

次に、ステップ ST 3 で、トランザクションや命令処理の解析を行なう。このとき、図 6 に示す動作シミュレーションを行なって、図 7 に示すトランザクション解析と、図 8 に示す命令処理解析とを行なって、挿入すべき FIFO の段数や必要なクロスバスの配置などを行なう。

そして、ステップ ST 4 で、例えば上述の 3 つの手法 (図 10, 図 11 など参照) による解析指標の判断を行なう。このステップ ST 1 ~ ST 4 の処理を繰り返し行なって、最適指標 O_p が最小のシステムを選択する。このステップ ST 3, ST 4 においては、ハードウェアとソフトウェアとを併せた性能評価 (HW/SW 性能評価) を行なうのである。

次に、ステップ ST 5 ~ ST 7 で、最適なシステムをハードウェアとソフトウェアとに分割する。すなわち、ステップ ST 5 で、最適 I F の合成を行ない、ス

ステップST.6で、図12に示すようなシステム接続図を生成する。一方、ステップST.7で、アプリケーション、フロー制御（FLOW）、OS（オペレーションシステム）などのソフトウェアの選択（条件設定）を行なう。

最後に、ステップST.8で、ハードウェア／ソフトウェアの協調検証を行なう。つまり、ハードウェアを用いてソフトウェアがうまく機能するかどうかを検証するのである。

本実施形態のインターフェースの設計方法によると、アプリケーションについてバス構造ごとに登録された動作モデルとなるライブラリを用いて動作シミュレーションを行ない、この動作シミュレーションの結果から求められる副パラメータ及び主パラメータに基づいて、インターフェースによって接続されるシステム全体の性能を実動作に近い環境で正確に評価することができる。そして、この評価に基づいて、設計者の要望にもっとも適したインターフェースを選択することができ、システム全体の構築に供することができる。

クレーム

1. 半導体集積回路中の制御回路と複数の被制御回路とを接続するためのバス構造であって、

上り側バスと下り側バスとに分離されたアドレスバスと、

上り側バスと下り側バスとに分離されたデータバスと、

を備えていることを特徴とするバス構造。

2. クレーム1のバス構造において、

上記データバスは、上記被制御回路ごとに分離され、かつ、被制御回路ごとに上り側バスと下り側バスとに分離されている。

3. 半導体集積回路の設計のために利用されるデータベースであって、

制御機能部と複数のアプリケーションとの間を接続するバス構造の種類に関する記述が含まれたテーブルを備えているデータベース。

4. クレーム3のデータベースにおいて、

上記テーブルは、各アプリケーションの動作モデルによって実現される性能を評価するための性能指数を記述した性能テーブルを備えている。

5. クレーム4のデータベースにおいて、

上記性能テーブルは、上記性能指数として、処理能力、バス幅、命令量及びメモリ量のうち少なくともいずれか1つのパラメータを含む。

6. クレーム3のデータベースにおいて、

上記性能テーブルは、バス構造に関する記述として、上り側バスと下り側バスとに分離されたアドレスバスと、上り側バスと下り側バスとに分離されたデータバスとからなる分離型バス構造に関する記述を含んでいる。

7. 複数のアプリケーションの各々に対してその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、

上記複数のアプリケーションの動作モデルとして上記複数のライブラリを順次用い、

上記各ライブラリについて、上記制御機能部により各アプリケーションを無制

限に動作させる動作シミュレーションを行なって、バストランザクションの衝突回数を解析する。

8. クレーム7のインターフェースの設計方法において、

各アプリケーション同士の間に上記トランザクションの衝突回数に応じた段数のFIFOを生成し、FIFOを仮挿入した状態で上記解析を行なう。

9. 複数のアプリケーションの各々に対してその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって

上記複数のアプリケーションの動作モデルとして上記複数のライブラリを順次用い、

上記各ライブラリについて、上記制御機能部により各アプリケーションを無制限に動作させる動作シミュレーションを行なって、同時命令処理の回数を解析する。

10. クレーム9のインターフェースの設計方法において、

上記同時命令数に応じてクロスバスの構成を決定する。

11. クレーム10のインターフェースの設計方法において、

さらに、上記同時命令数が所定値以上であるバスに配置するための転送動作の制御機能部を生成し、転送動作の制御機能部を配置した状態で上記解析を行なう。

12. 複数のアプリケーション及び複数のバス構造の各々に対してその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、

上記半導体集積回路の価値を最終的に評価するための複数の主パラメータを設定し、上記主パラメータに対して影響を与える複数の副パラメータを設定するステップ(a)と、

上記各主パラメータごとに、各ライブラリの副パラメータによって主パラメータを評価して、主パラメータの目標値を満たすライブラリ群を選び出すステップ

(b) と、

上記選び出されたライブラリ群ごとに定まる複数の主パラメータを評価して、最適なライブラリ群を選ぶことにより、インターフェースを決定するステップ (c) と

を含むインターフェースの設計方法。

13. クレーム 1 2 のインターフェースの設計方法において、

上記ステップ (a) の前に、上記複数のアプリケーションの動作モデルとして上記複数のライブラリを順次用いて、上記各ライブラリについての動作シミュレーションを行なうことにより、上記各ライブラリの副パラメータを解析するステップをさらに含む。

14. クレーム 1 2 のインターフェース設計方法において、

上記ステップ (a) では、3つの主パラメータを設定し、各主パラメータごとに3つの副パラメータを設定して、

上記ステップ (b) では、3つの副パラメータを座標軸とする3次元座標系を構成して、各副パラメータの値によって定まる3角形の面積が目標値以下のライブラリ群を選び出し、

上記ステップ (c) では、3つの主パラメータを座標軸とする3次元座標系を構成して、選び出されたライブラリ群から求まる主パラメータの値によって定まる3角形の面積が最小のライブラリ群からインターフェースを決定する。

15. クレーム 1 2 のインターフェースの設計方法において、

上記ステップ (a) の後ステップ (b) の前に、上記複数の副パラメータのうち特定の副パラメータに着目して、特定の副パラメータが目標値を満たしているライブラリ群を選び出し、

上記ステップ (b) では、上記複数の主パラメータのうち特定の主パラメータを除く主パラメータが目標値を満たしているライブラリ群を選び出し、

上記ステップ (c) では、上記特定の主パラメータが最小となるライブラリ群を最適なライブラリ群として選び出す。

16. クレーム 1 2 のインターフェース設計方法において、

上記ステップ (a) では、主パラメータに対する複数の副パラメータの影響係

数を設定し、

上記ステップ (b) では、影響係数と副パラメータの値とに基づいて各主パラメータの目標値を満たすライブラリ群を選び出し、

上記ステップ (b) では、選び出されたライブラリ群から求まる複数の主パラメータに重み付けを行なってから、ライブラリ群を決定するための演算を行なう。

17. 複数のアプリケーション及び複数のバス構造の各々に対してその動作モデルとなる複数のライブラリを記憶したデータベースを用いて、半導体集積回路の制御機能部と複数のアプリケーションとを接続するためのインターフェースを設計する方法であって、

上記複数のアプリケーションの動作モデルとして上記複数のライブラリを順次選択するステップ (a) と、

上記制御機能部により上記各アプリケーションを動作させて、各ライブラリを用いたときの制御機能部、インターフェース及びアプリケーションの性能を解析するステップ (b) と、

上記ステップ (a) とステップ (b) とを繰り返し行なって、上記解析結果に基づいて最適なライブラリ群を選ぶことにより、インターフェースを決定するステップ (c) と、

上記決定されたパラメータに基づいて最適なインターフェースを合成するステップ (d) と

を含むインターフェースの設計方法。

18. クレーム 17 のインターフェースの設計方法において、

上記ステップ (b) では、上記各ライブラリについて、上記制御機能部により各アプリケーションを無制限に動作させたときのバストランザクションの衝突回数を解析し、

上記ステップ (d) では、各アプリケーション同士の間に上記トランザクションの衝突回数に応じた段数の F I F O を挿入する。

19. クレーム 17 のインターフェースの設計方法において、

上記ステップ (b) では、上記各ライブラリについて、上記制御機能部により

各アプリケーションを無制限に動作させたときの同時命令処理の回数を解析し、
上記ステップ (d) では、上記同時命令数が所定値以上であるバスにはクロス
バスを配置する。

要 約

アプリケーションごとに、ノイマン型CPU（バス構造），ハーバード型CPU（バス構造），方向分離型CPU（バス構造）を採用したときの動作を記述した動作モデルとなるライブラリが登録されている。そして、各ライブラリの性能テーブルにおいては、各ライブラリの性能指標が、処理能力，バス幅，命令量，メモリ量というパラメータの関数として表されている。動作のうちソフトの部分とハード化された部分も登録されている。各アプリケーションを順次各ライブラリで置換して動作シミュレーションを行なうことで、半導体集積回路の性能が評価でき、最適なインターフェースを合成することができる。